Raphaël BEAU
Maxim LALANE
TP6

UPEC
Connaissance - Action
UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE

Compte-rendu SAE - DEV1.1

Sommaire :

I. Description du projet

Le jeu du serpent est un classique intemporel des jeux vidéo, célèbre pour sa simplicité et son addictivité. Dans ce jeu, le joueur contrôle un serpent qui se déplace à l'intérieur d'une grille. L'objectif principal est de faire croître le serpent en le faisant manger, tout en évitant les obstacles et les murs et en s'abstenant de se mordre la queue. Le tout est chronométré, on peut donc défier nos amis.

II. Les différentes fonctionnalités

Le serpent se déplace sur un plateau. Il peut aller vers le haut, vers le bas, vers la droite et vers la gauche, et ce grâce aux flèches directionnelles. En appuyant sur la touche <espace>, vous pouvez mettre en pause, et avec la touche <échap>, vous mettez fin à la partie. En ramassant des boules rouges, vous grandissez et augmentez votre score (chaque boule rouge vaut 5 points). Évitez les murs ou les boules violettes qui les font apparaître. Enfin, le temps défile à partir du début de la partie. Alors dépêchez-vous pour battre votre record !

III. Organisation du code/ diagramme

A. Découpage du code

Le code est découpé en 7 fichiers “.c”. Nous les avons rangés de façon ordonnée pour avoir un programme final le plus clair possible. En voici le découpage :

- Chemin.c contient les fonctions : Semage(), VerifChemin(), -VoirCase(), TempsArret() et Gagne().

Ces fonctions sont ensembles car elles servent principalement à vérifier les valeurs dans le tableau. Seuls TempsArret() et Semage() sont un peu différentes. Semage() écrit dans le tableau mais cette fonction sert avant tout à poser des balises pour que la queue puisse savoir dans quelle direction aller. TempsArret() permet de changer le timer de la queue pour faire grandir le serpent.

- Deplacement.c contient les fonctions : CliqueTouche(), DeplacementQueue() et DeplacementTete().

Ces fonctions sont ensemble car elles gèrent les déplacements du serpent. Les deux dernières écrivent beaucoup dans le tableau de façon à faire avancer respectivement la tête et la queue. CliqueTouche() sert quant à elle à tester les touches que le joueur entre sur son clavier. Cette fonctionnalité étant étroitement liée aux déplacements du serpent, nous avons décidé de la mettre dans ce même fichier.

- Afficher.c contient les fonctions : AfficheTab(), Affiche(), DessinerScore() et enfin DessinerTimer().

Ces fonctions sont ensemble car elles sont toutes concernées par l’affichage d’éléments en tous genres. Comme leurs noms l’indique, AfficherTimer() et AfficherScore() servent à afficher respectivement le temps de la partie et le score. Affiche() parcourt le tableau de valeurs et transmet les informations a AfficheTab() qui s’occupe d’afficher les bons éléments aux bonnes coordonnées.

- ModifTab.c contient les fonctions : init() et Pastille().

Ces fonctions sont ensemble car elles servent à initialiser et modifier toute la partie graphique, c'est-à-dire le plateau, les murs, les pastilles, et bien sûr le serpent. Pastille() ajoute des éléments d'un type choisi (murs,pommes…) et d'une quantité choisie. Init() initialise le tout.

- Menu.c contient les fonctions : MenuDebut(), MenuFinGagne(), MenuFinPerdu() et InitEcran().

Ces fonctions sont ensemble car, comme le nom du fichier l'indique, elles gèrent les menus. InitEcran() permet d'initialiser et de créer les fenêtres où le jeu apparaîtra. MenuDebut() sert à afficher le menu de départ. Il y a deux menus de fin : si on gagne ou si on perd, respectivement MenuFinGagne() et MenuFinPerdu().

- Jeu.c contient la fonction LancerJeu().

Ce fichier sert simplement à lancer le jeu pour pouvoir le rappeler (dans les menus de fin).

- main.c ne contient pas de fonctions. Ce fichier sert seulement à appeler LancerJeu() une première fois.

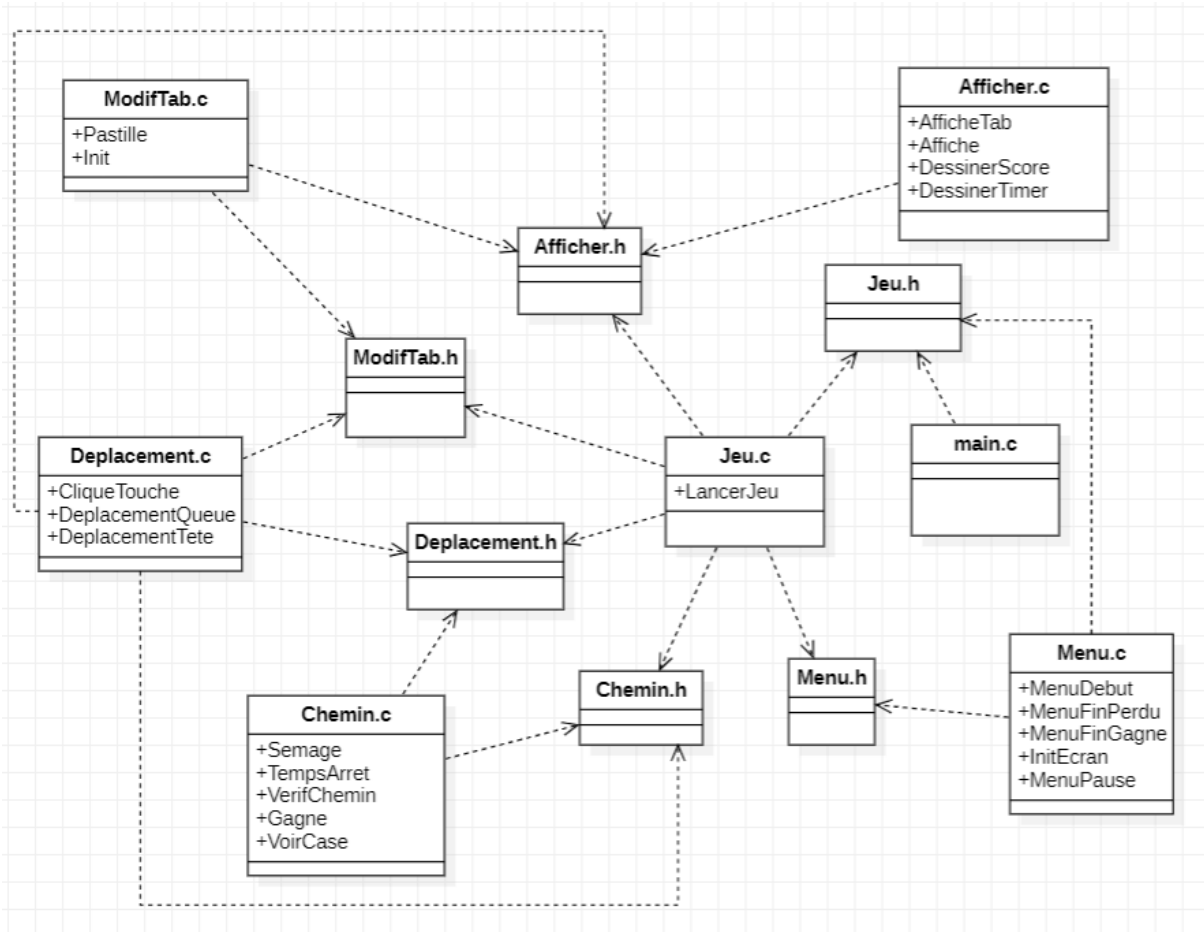
B. Appels entre fonctions

Pour le bon fonctionnement du programme, les fichiers s'appellent entre eux via la multitude de fonctions qui les composent. Ces appels sont listés ci-dessous :

- La fonction Semage() du fichier Chemin.c appelle la fonction CliqueTouche() du fichier Deplacement.c .
- La fonction DeplacementQueue() du fichier Deplaecement.c appelle la fonction VoirCase() du fichier Chemin.c
- La fonction DeplacementTete() du fichier Deplacement.c appelle la fonction VerifChemin() du fichier Chemin.c . Elle appelle également la Pastille() du fichier ModifTab.c et la fonction DessinerScore() du fichier Afficher.c .

- La fonction LancerJeu() du fichier Jeu.c appelle les fonctions InitEcran(), MenuDebut(), MenuFinGagne(), MenuFinPerdu() et MenuPause() du fichier Menu.c . De plus, cette fonction appelle les fonctions init() du fichier ModifTab.c, Affiche() et DessinerTimer() du fichier Afficher.c, Semage() et Gagne() du fichier Chemin.c, et DeplacementQueue() et DeplacementTete() du fichier Deplacement.c .

C. Diagramme associé



IV. Explications plus poussées

A. Principe Général

1. Back-End

Notre programme repose entièrement sur un tableau. Nous pourrions proposer un jeu qui se joue uniquement à la ligne de commande, grâce à notre programme. La grille demandée fait 60 cases sur 40 cases ; nous créons et utilisons donc un tableau de cette dimension. Pour avoir les murs tout autour du plateau de jeu, nous augmentons de 2 ces deux dimensions, ce qui nous donne 62 sur 42. Pour mettre des murs, nous donnons aux coordonnées choisies la valeur 40 ; pour mettre des pommes, la valeur 30 ; la valeur 50 pour les malus (boules violettes) ; les valeurs 1,6,7,8,9 pour le serpent (voir partie IV-B) ; enfin, la valeur par défaut est le 0. Nous plaçons nous-mêmes le serpent de 10 segments en haut à gauche de la grille en mettant des 1 dans le tableau aux endroits choisis. Nous plaçons les pommes et les malus en fonction du mode de jeu.

2. Front-End

Le tableau est affiché de façon constante dans notre while. De ce fait, la page peut être redimensionnée sans aucun problème durant la partie de jeu. Pour l'afficher, le programme parcourt le tableau (double boucle for de la taille du tableau) et affiche les éléments graphiques en fonction de ce qu'il y voit, en incrémentant les coordonnées pour savoir où placer ces éléments (+20 a chaque fois sinon il affichera tous les éléments au même endroit). Ces éléments font tous la même taille, 20 sur 20 pixels, pour que le tout reste homogène et cohérent.
Par exemple :

- Les murs (la valeur 40) seront affichés avec un carré gris rempli (comportant un carré vide noir plus petit à l'interieur).
- Les pommes (la valeur 30) prendront la forme de boules rouges.

- Les boules violettes (la valeur 50) représentent les malus qui font apparaître des murs sur le terrain de jeu.
- Le serpent (les valeurs 1,6,7,8,9) est affiché avec des carrés oranges.
- Le terrain vide (la valeur 0) est affiché avec des carrés noirs.

B. Représentation du serpent dans la mémoire

Le serpent n'a pas de représentation véritable dans la mémoire. C'est principalement un affichage. Mes explications étant trop longues, je les diviserai en trois parties que voici ci-dessous.

1. La Tête

La tête est l'élément clé du serpent. En effet, c'est elle qui détecte la prochaine case. Si elle voit que celle-ci est un 0, un 50 ou un 30, elle y va et remplace la valeur de la case où elle était par un 1. Ensuite, si la valeur de sa nouvelle case n'est pas 0, la tête demande de réafficher quelque part un nouvel élément du même type (pastille, mur) pour remplacer celui qu'elle a "mangé". Quand elle mange une pastille rouge, la tête va aussi indiquer à la queue de s'arrêter un court instant, afin d'allonger le serpent. En effet, si la tête continue d'avancer alors que la queue est arrêtée, alors le serpent grandit. Une fois les bons timings trouvés, au contact d'une pastille rouge (valeur de 30), le serpent s'allonge de 2 segments, comme demandé.

2. Le Corps

Il n'est qu'une suite de 1 dans le tableau. Il est rétréci par la queue qui avance, mais est agrandi en même temps par la tête qui avance aussi. Finalement, le corps n'est là que pour faire le lien entre la tête et la queue et n'apporte rien au programme. Il est purement graphique.

3. La Queue

La queue avance au même rythme que la tête. Elle suit la trace de 1 laissée au préalable par la tête. Lorsqu'elle tombe sur une case qui n'est pas un 1 (donc soit 6, 7, 8 ou 9), elle sait que sa direction doit changer, et adapte alors sa direction en fonction du chiffre affiché. Par exemple, le 6 dit qu'il faut changer de direction pour aller vers le haut au prochain déplacement.
Lorsque la queue change alors de case, elle remplace la valeur de son ancienne case par un 0. Tout comme la tête laisse une ligne de 1 derrière elle, la queue laisse donc une ligne de 0 là où elle passe.
La tête, quand elle mange une pastille rouge, peut aussi différer le timing de la queue, de façon à la faire attendre le temps que la tête avance de deux segments pour faire grandir le serpent.

4. Les Déplacements

Mais concrètement, comment avance le serpent ? La réponse est grâce aux timers que nous avons produits avec la fonction Microsecondes(). En effet, tous les x temps (x défini à l'avance), des indices spécifiques du tableau sont modifiés. Cela a pour conséquence de faire bouger le serpent, étant donné que le tableau est affiché en permanence. Avec des variables que l'on incrémente en gardant une direction bien précise (en n'augmentant que la variable de l'axe des abscisses par exemple), le serpent bougera alors dans la direction que l'on souhaite.

V. Conclusion des membres du groupe

A. Maxim

Grâce ce premier projet de DEV, j’ai eu l’occasion de recréer en langage C une version de l’un des classiques du jeu-vidéo, le fameux “Snake”. Ce jeu a l’avantage d’être relativement abordable à coder, ce qui le rend accessible aux personnes comme nous qui n’avons pas encore une grande expérience en C. De plus, cela nous permet aussi de réaliser notre premier jeu en C, et ainsi d’avoir notre premier code que l’on va vraiment réutiliser dans la vie courante.
Lors de la réalisation de ce projet, nous nous sommes partagés les tâches avec mon binôme, et je me suis donc surtout occupé de l’affichage graphique et des menus. Je garde un très bon souvenir de ce projet, et j’en suis ressorti grandi et impatient de voir quelle sera la suite.

B. Raphaël

Cette année, j'ai eu l'opportunité de revisiter le projet Snake en langage C. Confronté aux défis du projet, j'ai concentré mes efforts sur l'implémentation des mouvements du serpent et la gestion des collisions.
J'ai particulièrement apprécié la résolution de problèmes, et la satisfaction de voir mon code générer un jeu fonctionnel. Cette expérience m'a apporté une compréhension plus approfondie du langage C, renforçant mes compétences en programmation.
Ce projet m'a enseigné la persévérance, transformant les obstacles en leçons constructives. Je suis fier des progrès réalisés, et je suis convaincu que cette expérience contribuera à ma croissance en tant qu'étudiant en informatique.

VI. Les différents modes de jeu

A. Normal

Le mode “Normal” représente le jeu “Snake” classique. Vous devez faire grandir votre serpent en faisant attention à ce qu’il ne se morde pas la queue ou qu’il ne percute pas un mur. Lorsque vous mangez une pomme, le serpent grandit de 2 segments et une nouvelle pomme apparaît aléatoirement sur le plateau de jeu.

B. Mirror

Le mode “Mirror” ajoute quelques spécificités au jeu. Tout d’abord, les murs sur les côtés du plateau de jeu (à gauche et à droite) disparaissent. Le serpent peut donc naviguer d’un bout à l'autre sans problème. En effet, le jeu étant régi par un tableau, il suffit d’incrémenter ou de décrémenter l’indice du serpent pour faire cela ; il réapparaîtra simplement à la ligne du dessous ou du dessus.
De plus, des pommes violettes font leur apparition, au nombre de 2 sur le plateau. A chaque fois que le serpent en mange une, ses points n’augmentent pas, et il ne grandit pas non plus, mais 5 murs apparaissent aléatoirement sur le plateau.

C. Hardcore

Le mode “Hardcore” porte très bien son nom. En effet, dans ce mode de jeu, la queue ne remplace pas les valeurs derrière elle par des 0 mais par des 40, ce qui a pour effet de semer des murs dans le sillage du serpent. De plus, les boules violettes font désormais apparaître 20 murs au lieu de 5. Les murs qui avaient disparu dans le mode “Mirror” reviennent également, afin de ne laisser aucun échappatoire aux joueurs. Bien que ce mode n'ait pas vraiment été conçu pour être terminé, il est tout à fait possible de le faire et cette éventualité est prévue dans le programme. Toute partie peut être gagnée ou perdue, et ce dans n’importe quel mode de jeu.